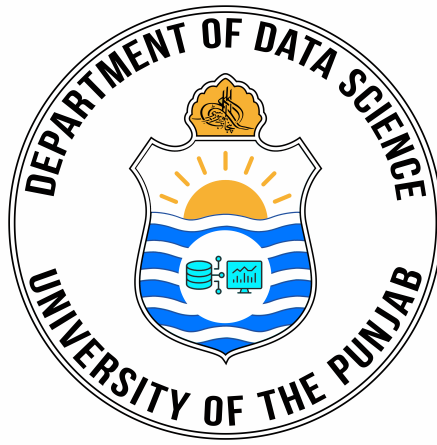


Final Year Project Proposal

LLMShield

A Unified Threat Detection Framework for Mitigating Prompt Injection,
Model Poisoning, and RAG Embedding Risks



By

Alisha Shahid BSDSF22A005

Um-e-Abeeha BSDSF22A009

Khalood Sami BSDSF22A021

Under the supervision of

Dr. Muhammad Arif Butt

Bachelor of Science in Data Science (2022-2026)

FACULTY OF COMPUTING INFORMATION TECHNOLOGY (FCIT),

UNIVERSITY OF THE PUNJAB, LAHORE

Chapter 3

User Stories & Epics

3.1 Epic: User Authentication and Authorization

Title: Account Lifecycle & Credential Management
Priority: High
User Story: As a user, I want to securely register, verify my identity, log in, reset my password, and manage my profile, so that I can maintain full control of my account throughout its lifecycle and protect my personal information.
Acceptance Criteria: <ul style="list-style-type: none">• Given a user registers with a unique email & strong password, When they submit the registration form, Then the system creates an unverified account, sends a verification email, and prevents login until verification is complete.• Given a user clicks a valid verification link, When the system validates the token, Then the account status changes to verified, and the user is redirected to the login page.• Given a verified user enters correct credentials, When authentication succeeds, Then the system issues a secure, time-bound session token (e.g., JWT with refresh token) and redirects to the dashboard.• Given a user requests a password reset, When they access a valid reset link and submit a new password meeting security requirements, Then the old password is invalidated and active sessions are terminated,• Given a logged-in user updates profile details or changes their password, When they confirm with their current password, Then the changes are applied immediately and a notification is seen if credentials or security settings were updated.

Table 3.1. Account Lifecycle & Credential Management

Title: Multi-Factor Authentication (MFA)
Priority: High
<p>User Story: As a security-conscious user, I want to enable and manage multi-factor authentication (MFA) using a TOTP authenticator app and recovery codes, so that I can add a critical second layer of security to my account, protecting it even if my password is compromised.</p>
<p>Acceptance Criteria:</p> <ul style="list-style-type: none"> • Given a logged-in user navigates to their “Security Settings” page, When they initiate MFA enrollment, Then the system must present a unique QR code compatible with standard TOTP applications (e.g., Google Authenticator, Authy). • Given a user scans the QR code and correctly enters the corresponding 6-digit TOTP code, When they confirm the setup, Then MFA must be activated and a set of single-use recovery codes generated and presented for secure storage. • Given MFA is enabled, When a user logs in, Then they must provide both their password and a valid TOTP or unused recovery code before access is granted. • Given MFA is enabled, When an invalid, expired, or already used TOTP/recovery code is entered, Then login must be denied and the failed attempt logged for auditing. • Given MFA is enabled, When a user attempts to disable MFA, Then they must confirm the action using both their password and a valid TOTP code, after which MFA is deactivated. • Given MFA is enabled, When a user selects “Trust this device for 30 days”, Then the system must set a secure, long-lived cookie that reduces login friction on that device only. • Given MFA is enabled, When a user attempts to log in without providing a second factor, Then access must be denied. • Given recovery codes are generated, When one is used, Then it must be immediately invalidated and cannot be reused.

Table 3.2. Multi-Factor Authentication (MFA)

3.2 Epic: Model Configuration & Setup

This epic allows users to configure LLM models, apply custom rules, integrate with external tools, and save/load configuration profiles. Ensure flexible and consistent setup for scanning workflows.

Title: Configure LLM Model Parameters
Priority: Medium
User Story: As a user, I want to configure LLM model parameters (e.g., type, temperature, max tokens, top-p), so that I can customize how the system runs scans.
Acceptance Criteria: <ul style="list-style-type: none">• Given a user opens configuration, When they select a model type (e.g., GPT, Claude, LLaMA), Then the choice is saved and applied to scans.• Given a user updates parameters (temperature, max tokens, top-p), When values are valid, Then the system accepts and applies them.• Given a user provides invalid input (negative values, exceeding max tokens), When they attempt to save, Then clear error messages appear.• Given no custom configuration is provided, When the user proceeds, Then default settings are automatically loaded.• Given valid changes are saved, When a scan runs, Then the configuration is applied consistently.

Table 3.3. Configure LLM Model Parameters

Title: Save & Load Model Configurations
Priority: Medium
User Story: As a user, I want to save, load, and manage configuration profiles, so that I can reuse preferred model setups easily.
Acceptance Criteria: <ul style="list-style-type: none"> • Given a user saves current settings, When the profile is created, Then it appears in the dashboard with creation date. • Given a user selects a saved profile, When it is loaded, Then all settings update instantly. • Given a profile is no longer needed, When the user deletes it, Then it is removed permanently. • Given duplicate profile names exist, When the user tries to save, Then the system prevents duplication or asks for overwrite confirmation. • Given a user logs back in, When they check profiles, Then saved profiles persist securely across sessions and accounts.

Table 3.4. Save & Load Model Configurations (Profiles)

3.3 Epic: Prompt Injection Detection

This epic focuses on detecting and reporting prompt injection risks in prompts, models and docs. It ensures adversarial instructions and malicious payloads are identified and mitigated.

Title: Model Probe-based Injection Scan
Priority: High
User Story: As a security analyst, I want to run predefined adversarial probes against a model so I can detect policy-breaking behavior.
<p>Acceptance Criteria:</p> <ul style="list-style-type: none"> • Given a model is selected, When the scan starts, Then the system runs each probe in a fresh isolated session and records prompt, response, latency, and token usage. • Given a probe response matches violation rules (e.g., followed adversarial instruction, leaked hidden/system info, executed disallowed action), When execution completes, Then the system flags it with rule ID, confidence score, and severity. • Given a probe errors or times out, When execution completes, Then the result is marked Error/Timeout with reason and does not block remaining probes. • Given the scan finishes, When results are compiled, Then a summary shows probes total, passed/failed/errored counts, and top violating categories.

Table 3.5. Model Probe-based Injection Scan

Title: Document Injection Scan
Priority: High
User Story: As a developer/analyst, I want RAG documents scanned for embedded malicious instructions so I can prevent retrieval-time prompt injection.
<p>Acceptance Criteria:</p> <ul style="list-style-type: none"> • Given supported documents (e.g., PDF, DOCX, MD, TXT) are uploaded, When a scan runs, Then the system extracts text and analyzes for instruction-like payloads. • Given suspicious content is detected, When results are returned, Then each finding includes type, severity, snippet, and location (file + page/line/offset/chunk). • Given no suspicious content is found, When results are returned, Then the system shows “No injection indicators found” with coverage stats. • Given a finding exists, When the user views remediation, Then the system provides guidance (sanitize, strip role-like prefixes, escape, allow/deny lists) with config links. • Given files exceed limits, When upload/scan is attempted, Then the system shows clear size/type limits and supports batching.

Table 3.6. Document Injection Scan

Title: RAG Retrieval Injection Simulation
Priority: Medium
User Story: As a security analyst, I want to simulate retrieval and inference using my RAG config to detect combined injection paths.
<p>Acceptance Criteria:</p> <ul style="list-style-type: none"> • Given a RAG configuration and document set, When simulation runs on sample queries, Then the system executes retrieval, builds final prompt, and invokes the model. • Given model output exhibits instruction-following from retrieved text, When results are generated, Then the system flags injection success and links offending snippets to output. • Given simulation completes, When user views results, Then they see pass/fail per query, severity, retrieved source IDs, and replay options. • Given retrieval or inference fails, When errors occur, Then the failure is logged with component, error type, and simulation continues.

Table 3.7. RAG Retrieval Injection Simulation

Title: Report & Explain Injection Findings
Priority: High
User Story: As a user, I want clear descriptions, risk levels, and mitigation steps for each finding so I can prioritize fixes and document compliance.
<p>Acceptance Criteria:</p> <ul style="list-style-type: none"> • Given findings exist, When the user opens a report, Then each item shows title, description, category, severity rubric, evidence, and reproducibility info. • Given user needs to act, When viewing a finding, Then the report provides concrete mitigations with links to relevant settings. • Given compliance docs are required, When exporting, Then system generates PDF and JSON/CSV with all findings, metadata, and audit trail. • Given many findings exist, When filtering/sorting, Then users can filter by severity, category, asset, and timeframe, and bookmark/share filtered views. • Given a finding is resolved, When re-scan shows no issue, Then the report marks it “Remediated” linking old → new scan IDs.

Table 3.8. Report & Explain Injection Findings

3.4 Epic: Model Poisoning Detection

This epic focuses on detecting and mitigating model poisoning risks through behavioral anomaly testing, trigger/targeted-output tests, and audit-ready evidence. It ensures poisoned or tampered models are identified, validated, and documented for compliance.

Title: Behavioral Anomaly Detection
Priority: High
User Story: As a security analyst, I want input-output tests that reveal inconsistent or target-specific malicious behavior so I can detect potential poisoning.
<p>Acceptance Criteria:</p> <ul style="list-style-type: none"> • Given a baseline is defined (reference model, prior version, or gold answers), When a test suite is created, Then it includes neutral controls, targeted classes/entities, and negative controls. • Given the suite runs with fixed parameters, When outputs are collected, Then anomaly scores (e.g., target flip rate, stance shift, policy-violation rate) are computed vs baseline and deviations threshold (e.g., 25%) are flagged. • Given anomalies are detected, When they cluster on specific entities/keywords/-classes, Then findings are marked “target-specific” with targets listed. • Given a finding is flagged, When repeated N times or across two temperatures, Then system reports consistency (70%) or downgrades severity if not reproducible. • Given a probe errors or times out, When the suite completes, Then errors are logged per item without aborting the run. • Given results are ready, When viewed, Then a summary shows probes run, flagged count, targets impacted, severity per category, and replay links.

Table 3.9. Behavioral Anomaly Detection

Title: Trigger / Targeted-Output Tests
Priority: High
User Story: As a developer, I want tests that probe whether specific triggers produce harmful or unexpected outputs so I can validate whether poisoning payloads are present.
<p>Acceptance Criteria:</p> <ul style="list-style-type: none"> • Given a trigger library (keywords, Unicode variants, prefixes/suffixes) and custom triggers, When a plan is saved, Then both sets are included and versioned. • Given a trigger is tested K times, When harmful/targeted behavior appears in T% of trials (e.g., 60%), Then the trigger is flagged with severity mapped to reliability and impact. • Given multilingual/obfuscated variants exist, When the suite runs, Then near-duplicate patterns (homoglyphs/spaces) are generated and tested. • Given a response is produced, When logging occurs, Then prompt, trigger form, response, and classifier tags (toxicity/PII/leakage) are stored. • Given negative controls exist, When runs complete, Then the false-positive rate is reported and factored into severity.

Table 3.10. Trigger / Targeted-Output Tests

Title: Model Audit Snapshot & Evidence
Priority: Medium
User Story: As a compliance officer or analyst, I want audit-ready evidence of poisoning scans so I can support investigations or audits.
<p>Acceptance Criteria:</p> <ul style="list-style-type: none"> • Given a scan completes, When metadata is recorded, Then report includes model ID/version/hash, config hash, seed/params, suite version, operator, timestamps, and environment. • Given flagged findings exist, When exported, Then package contains raw input-s/outputs, trigger variants, anomaly scores, thresholds, and rationale; sensitive token-s/keys are redacted. • Given export is requested, When generated, Then both human-readable (PDF) and machine-readable (JSON/CSV) formats are produced with stable IDs. • Given the bundle is created, When finalized, Then a checksum/signature is included for tamper evidence. • Given retention policies apply, When period elapses, Then artifacts expire with audit log entry.

Table 3.11. Model Audit Snapshot & Evidence

3.5 Epic: Embedding & Vector Security Analysis

Title: Document-to-Embedding Inspection
Priority: High
User Story: As a developer/analyst, I want to scan documents for content that would produce adversarial/poisonous embeddings so I can prevent harmful entries in the vector store.
Acceptance Criteria: <ul style="list-style-type: none">• Given a supported document (PDF/DOCX/MD/TXT) is uploaded, When analysis runs, Then the system extracts text using the same chunking/settings used by the embedding pipeline.• Given the text is analyzed, When suspicious patterns are found (e.g., instruction-like payloads, trigger phrases, obfuscated tokens, extreme repetition), Then the system flags specific passages with: snippet, location (file + page/line/chunk), reason label, and a risk score.• Given flagged passages exist, When the user views remediation, Then the system recommends concrete actions (e.g., sanitize/remove/mask, adjust chunk size/overlap, stopword list, denylist patterns) and offers a preview of post-sanitization chunks.• Given a user applies sanitation or excludes chunks, When they save, Then the analysis re-runs on the affected chunks and updates findings.• Given a file exceeds limits or fails to parse, When processing occurs, Then the system reports the error (type, size/format) and does not silently truncate.

Table 3.12. Document-to-Embedding Inspection

Title: Vector Store Anomaly Detection
Priority: Low
User Story: As a security analyst, I want the scanner to analyze the vector DB for suspicious vectors, high-similarity collisions, or injection-like entries so retrieval integrity is verifiable.
<p>Acceptance Criteria:</p> <ul style="list-style-type: none"> • Given a read-only snapshot of the vector index (embeddings + IDs + metadata) is available, When analysis runs, Then the system computes distribution and neighborhood stats (norms, density, collision rates) without modifying the store. • Given anomalies exist, When results are returned, Then the system flags: <ul style="list-style-type: none"> – dense clusters spanning unrelated sources/tenants (possible poisoning), – high-similarity collisions across different labels/topics, – extreme-norm/outlier vectors, – vectors tied to known trigger patterns (where metadata/text available). • Given items are flagged, When the user opens details, Then each finding includes vector/record ID, source doc/chunk, nearest neighbors, similarity scores, category, confidence, and recommended actions (e.g., quarantine records, re-embed with new model, tighten metadata filters). • Given large stores, When analysis is configured, Then sampling/batch modes can be selected and the report indicates coverage and confidence. • Given multi-tenant setups, When cross-tenant collisions are detected, Then the system flags potential data-partition leakage with tenant IDs.

Table 3.13. Vector Store Anomaly Detection

Title: Retrieval Attack Simulation
Priority: Medium
User Story: As an analyst/developer, I want to simulate adversarial queries that manipulate retrieval ranking or return poisoned passages so I can measure downstream impact.
Acceptance Criteria: <ul style="list-style-type: none"> • Given a set of queries (seeded or user-provided) and an index, When simulation runs, Then the system executes retrieval (top-k, filters, re-ranking) and logs baseline vs adversarial variants (paraphrase/Unicode/homoglyph/trigger-augmented). • Given retrieval results are produced, When rankings differ materially (e.g., target chunk moves into top-k or rank improves by configurable threshold), Then the system flags a retrieval manipulation with before/after rankings and responsible vectors/doc chunks. • Given model inference is enabled, When retrieved chunks are fed to the model, Then the system reports behavioral impact (e.g., policy violation, topic flip, toxicity/PII classifiers) and links outputs to specific retrieved chunks (trace: query → chunks → composed prompt → response). • Given errors/timeouts occur, When a run completes, Then failures are logged per query without aborting the whole suite and are excluded from success-rate metrics. • Given findings exist, When the user exports, Then a report includes ASR (attack success rate), rank-shift metrics, implicated vector IDs, and reproducible parameters (index version, embedding model, k, similarity metric).

Table 3.14. Retrieval Attack Simulation

3.6 Epic: RAG Pipeline Security

This epic provides both secure and insecure RAG pipeline setups, allows simulation of end-to-end attacks (retrieval → context assembly → inference), and enables users to harden pipelines. It ensures hands-on understanding of RAG vulnerabilities and ways to mitigate them.

Title: Provision Secure & Insecure RAG Templates
Priority: High
User Story: As a developer, I want pre-built secure and intentionally insecure RAG pipeline templates so that I can test the difference and demonstrate mitigation effects.
Acceptance Criteria: <ul style="list-style-type: none"> • Given I choose a template, When I deploy it in the sandbox, Then the pipeline is instantiated with documented components (retriever, vector DB, context assembly rules) and labeled as secure/insecure.

Table 3.15. Provision Secure & Insecure RAG Templates

Title: End-to-End RAG Attack Simulation
Priority: High
User Story: As a security analyst, I want to run end-to-end simulations that combine document injection, embedding manipulation, and model probes so that I can observe realistic attack chains and mitigations.
Acceptance Criteria: <ul style="list-style-type: none"> • Given a RAG pipeline and test dataset, When the simulation runs, Then the system outputs which stage(s) succeeded in causing malicious/inappropriate model outputs and recommends mitigations per stage.

Table 3.16. End-to-End RAG Attack Simulation

Title: Hardening Recommendations for RAG
Priority: Medium
User Story: As a compliance officer or developer, I want prescriptive hardening steps (retrieval filtering, sanitization, response filtering) for my RAG pipeline so that I can reduce real-world risk.
Acceptance Criteria: <ul style="list-style-type: none">• Given RAG simulation results, When I view the hardening section, Then I receive prioritized, actionable mitigation steps mapped to pipeline components.

Table 3.17. Hardening Recommendations for RAG

3.7 Epic: C/C++ Code & Repository Security Scanner

Provide static analysis of C/C++ codebases (uploaded archives or GitHub repos) to detect memory safety flaws, unsafe APIs, and hardcoded secrets, with machine-readable exports.

Title: Code Submission & Vulnerability Analysis
Priority: High
User Story: As a developer, I want to upload a code archive or link a GitHub repo so the system can run static analysis for memory safety flaws and unsafe functions.
<p>Acceptance Criteria:</p> <ul style="list-style-type: none"> • Given a user uploads .c/.cpp files or repo link, When scanning begins, Then the system clones/parses the project and identifies insecure function usage (e.g., <code>strcpy</code>, <code>gets</code>, <code>sprintf</code>). • Findings include function name, line number, CWE reference (e.g., CWE-120: Buffer Overflow, CWE-676: Use of Potentially Dangerous Function), safer alternative suggestions, and severity rating. • If build files are present, the system supports optional build-aware analysis (Make/C-Make). • Results must be viewable in the UI with filtering (by severity, CWE, file).

Table 3.18. Code Submission & Vulnerability Analysis

Title: Hard-coded Secrets & Credential Detection
Priority: Medium
User Story: As a security conscious developer, I want the scanner to detect hardcoded secrets so I can prevent accidental credential leaks.
Acceptance Criteria: <ul style="list-style-type: none"> • Given a codebase is scanned, When entropy/regex checks run, Then the system detects secrets such as AWS keys, JWTs, SSH keys, API tokens. • Findings are labeled as Critical severity with remediation advice (e.g., “Remove and store in environment variables”). • Each finding must be mapped to CWE references (e.g., CWE-798: Use of Hard-coded Credentials, CWE-321: Use of Hard-coded Cryptographic Key). • Detected secrets must be redacted in reports (only partial shown, e.g., last 4 chars).

Table 3.19. Hardcoded Secrets & Credential Detection

Title: Machine readable Reports
Priority: Low
User Story: As a developer/compliance officer, I want structured exports so I can integrate scan results into CI pipelines and audits.
Acceptance Criteria: <ul style="list-style-type: none"> • Given scan results exist, When I request export, Then results are available in JSON (primary) and CSV (optional) with fields: repo/commit, findings array (file, line, CWE, severity, description, remediation). • Given integration is configured, When a scan completes, Then results can be pushed to CI/CD (exit code or webhook) with gating based on severity thresholds. • Reports must include CWE mappings for each finding, along with metadata (job id, scanner version, timestamp, commit hash).

Table 3.20. Machine readable Reports

3.8 Epic: Reporting, Analytics & Compliance Exports

Title: PDF & Machine-Readable Exports
Priority: High
User Story: As a security analyst, I want to export scan results in PDF, JSON, and CSV formats so that I can share findings with both technical and non-technical stakeholders.
Acceptance Criteria: <ul style="list-style-type: none">• Given scan results exist, When I request an export, Then the system generates:<ul style="list-style-type: none">– PDF: Executive summary, technical findings, remediation steps, evidence appendix.– JSON/CSV: Structured findings with metadata (scan id, timestamp, severity, CWE, remediation).• Reports are versioned, digitally signed, and downloadable.

Table 3.21. PDF & Machine-Readable Exports

Title: Trend Analytics & Dashboards
Priority: Medium
User Story: As an admin or analyst, I want trend dashboards so I can monitor security posture over time.
Acceptance Criteria: <ul style="list-style-type: none"> • Given historical scan data exists, When I view analytics, Then I can filter by model/team/risk type and see charts of: <ul style="list-style-type: none"> – Critical/High findings over time – Mean time to remediate – Scan coverage & frequency • Data is exportable as CSV/PDF.

Table 3.22. Trend Analytics & Dashboards

3.9 Epic : Chatbot Assistance & Guided Remediation

This epic provides an interactive assistant within the UI of scan results to explain findings, guide remediation, and support developer workflows.

Title: Explain Finding in Natural Language
Priority: High
User Story: As a developer, I want the chatbot to explain a finding in simple language so that I can understand its impact.
Acceptance Criteria: <ul style="list-style-type: none"> • Given a finding, When I ask “Explain this finding,” Then the chatbot provides: <ul style="list-style-type: none"> – A plain-language explanation – Why it matters (impact/risk)

Table 3.23. Explain Finding in Natural Language

Title: Guided Remediation Steps
Priority: High
User Story: As a developer, I want the chatbot to suggest remediation actions so that I can efficiently fix vulnerabilities.
Acceptance Criteria: <ul style="list-style-type: none"> • Given a finding, When I request remediation, Then the chatbot provides: <ul style="list-style-type: none"> – Step-by-step fixes (code/policy/config examples where relevant) – Prioritized by severity – References to internal/external documentation

Table 3.24. Guided Remediation Steps

Title: Export Chat Sessions for Audit
Priority: Low
User Story: As a compliance officer, I want to export chat sessions for audit trails so remediation discussions are recorded.
Acceptance Criteria: <ul style="list-style-type: none"> • Given a chat session occurred, When I export it, Then the transcript is downloadable (PDF/JSON) with metadata: scan ID, user, timestamps.

Table 3.25. Export Chat Sessions for Audit